Premiers pas en Fortran

Module d'informatique - Licence DSM ENS Lyon

TD n° 2 — 2002-2003

24 octobre 2002, y.copin@ipnl.in2p3.fr

Objectifs du TD : source, compilation & exécution, structure, déclarations

1 Méthodologie de programmation

L'utilisation d'un langage de programmation (p.ex. le Fortran) permet de générer un programme binaire exécutable par la machine (l'*exécutable*) à partir d'un simple fichier texte (un fichier « ASCII ») indiquant précisément et dans une syntaxe particulière la marche à suivre (le fichier *source*).

En général, on ne programme pas uniquemenent pour le plaisir, mais pour faire exécuter par la machine des tâches précises. La programmation peut se décomposer en quelques grandes étapes :

Analyse : il s'agit de bien préciser, dans un langage libre, ce que l'on veut que la machine fasse, et les méthodes générales à utiliser. P.ex. : « afficher la chaîne de caractères "*Hello world !*" dans le terminal ».

Pseudo-code : il s'agit maintenant de décrire en détail les différentes étapes du programme – l'*algorithme* – dans un langage formel. P.ex. :

```
Programme Hello
 afficher  "Hello world!"
fin
```

Codage : il faut alors traduire l'algorithme décrit précédemment dans le langage de programmation proprement dit, ici en Fortran. P.ex.[1] :

```
C2345␣Programme␣minimal
␣␣␣␣␣␣PROGRAM␣hello
␣␣␣␣␣␣PRINT*,"Hello␣world!"
␣␣␣␣␣␣END
```

[1] Le symbole « ␣ » représente simplement une espace, mais tous les autres caractères sont significatifs. Par ailleurs, les différences de polices (gras, italiques, etc.) ne sont que décoratives, puisque le source n'est qu'un simple fichier texte.

Debug : après avoir généré l'exécutable et l'avoir lancé sur la machine, il faut s'assurer que le programme satisfait bien aux objectifs définis dans la phase d'analyse, et éventuellement le corriger si ce n'est pas le cas.

> **Attention :** Ne pas négliger les deux premières étapes – l'analyse et le pseudo-code –, au risque de voir la phase de debug – de loin la plus fastidieuse – s'éterniser...

2 De la source à l'exécution

Même si les autres étapes ne sont pas à négliger, nous nous attardons maintenant sur la phase la plus technique de la programmation : le *codage*. En pratique, la réalisation et l'utilisation d'un programme Fortran s'articule en 3 étapes :

Codage : À l'aide de l'éditeur de texte de votre choix (p.ex. `emacs`), il faut créer un fichier texte contenant le programme en langage Fortran. Par convention, le nom du fichier portera l'extension `.f`.

Compilation : Il faut alors traduire le fichier source en un exécutable : c'est la *compilation*, réalisé par un programme spécifique, le *compilateur*. Pour un fichier source `prog.f`, on utilise la commande suivante dans un terminal :

```
g77 prog.f
```

ce qui produit – s'il n'y a pas d'erreur dans le code – un exécutable nommé par défaut `a.out`. Pour imposer un nom plus parlant, p.ex. `prog.exe`, on peut utiliser la commande suivante :

```
g77 -o prog.exe prog.f
```

Si la compilation n'aboutit pas, le compilateur vous indique l'endroit dans votre fichier source où il ne comprend pas la syntaxe ou la logique Fortran[2]. Il vous faut alors corriger les erreurs avant de relancer la compilation.

Exécution : Il s'agit maintenant d'exécuter votre programme dans le terminal, p.ex. à l'aide de la commande `./prog.exe`.

Si vous n'obtenez pas le résultat escompté ou si le programme s'interrompt à l'exécution avec un affichage du style `core dump`, alors les ennuis commencent : Il faut debugger votre programme, c.-à-d. identifier et corriger l'erreur *conceptuelle* (et non pas simplement de syntaxe) présent dans votre programme.

Exercice : Recopier *très précisément* – y compris les espaces – le code Fortran de la section précédente dans un fichier `hello.f`. Compiler et exécuter votre programme `hello.exe`. En êtes-vous satisfait ?

[2] Pour forcer le compilateur à vous avertir du moindre problème potentiel, ajouter l'option `-Wall` à la ligne de compilation : `g77 -Wall -o prog.exe prog.f`

3 Structure d'un programme Fortran

3.1 Format des lignes d'instruction

L'écriture d'un programme Fortran obéit à un « format carte » (fossile des temps anciens où les programmes étaient stockés sur cartes perforées...) :

- Une ligne de programme « standard » ne commence en fait qu'à la 7e colonne, les 6 premiers caractères étant remplis par des blancs (cf. exemple ci-dessous).
- Les instructions ne doivent pas dépasser la 72e colonne. Pour prolonger une instruction sur plusieurs lignes, il suffit d'introduire un caractère quelconque (généralement un « `*` ») à la 6e colonne de la ligne suivante. P.ex.

```
C␣Ceci␣est␣un␣commentaire,
C␣precedent␣une␣instruction␣s'etendant␣sur␣2␣lignes.
C234567
␣␣␣␣␣␣sum␣=␣sin(y)␣+
␣␣␣␣␣*␣␣␣␣␣␣cos(z)
C234567
```

- Dans la 1re colonne, la présence d'un caractère « `C` » ou « `*` » signale que la ligne est un commentaire : elle sera totalement ignorée par le compilateur.
- Les colonnes 1 à 5 servent à placer des « étiquettes » qui seront utilisées pour repérer une ligne. Elles sont utilisées p.ex. avec l'instruction `goto` ou pour les spécifications de formats d'entrée-sortie.
- Contrairement au *shell* utilisé dans vos terminaux,

  Attention : Le Fortran ne fait *aucune* distinction entre majuscules et minuscules.

  « `TOTO` », « `TotO` » et « `toto` » représentent *exactement* la même chose. Vous pouvez en profiter pour écrire en majuscules les mots-clés propres au Fortran.
- Évitez toute utilisation de lettres accentuées, y compris dans les commentaires et les chaînes de caractères.

3.2 Structure générale

Un programme Fortran se compose de deux parties : une partie *déclarative* (la « liste des ingrédients ») et une partie *exécutoire* (la « recette »).

La partie déclarative Tout programme Fortran commence par le mot `PROGRAM`, suivi du nom du programme (judicieusement choisi, sans espace ni accent).

Vient alors la partie *déclaration des variables*.

- Toute variable possède un type. Les types usuels sont les suivants :

  `INTEGER` nombre entier,

`REAL` nombre réel, simple précision,

`DOUBLE PRECISION` nombre réel, double précision,

`COMPLEX` nombre complexe,

`LOGICAL` variable booléenne,

`CHARACTER` chaîne de caractères (lettres, chiffres, symboles, etc.).

- Il est **hautement** préférable que toute variable soit *explicitement* déclarée – c'est à dire que son type doit être défini – dans l'unité dans laquelle elle sera utilisée. Pour forcer ce comportement,

  Ne pas oublier : Avant toute déclaration, utiliser la commande `IMPLICIT NONE`.

- Les variables ne sont pas initialisées automatiquement : leur valeur immédiatement après leur declaration est *quelconque*. Il faut donc bien penser à les initialiser lorsque c'est nécessaire.

C'est également dans la partie déclarative que l'on peut/doit ajouter les commentaires précisant l'auteur, la date de création du programme et sa version, ses objectifs et son mode d'emploi, ses limites, sa ligne de compilation, etc. (voir exemple ci-dessous).

La partie exécutoire Le *corps* du programme principal est encadré par les instructions de déclaration précédentes et `END`. Chaque ligne contient au maximum une instruction, certaines instructions pouvant s'étendre sur plusieurs lignes à l'aide du caractère de continuation en 6^{e} colonne. Il est de *très* bon ton de commenter un maximum son programme, afin d'expliciter son objectif, sa démarche, ses variables, etc.

La structure d'un programme Fortran est donc typiquement :

```
C23456
      PROGRAM nomduprogramme

C Commentaires sur le programme
C Date de creation, auteur, version
C Objectif, mode d'emploi, ligne de compilation, etc.

C Declaration explicite des variables
      IMPLICIT NONE
      declaration_1
      declaration_2
      ...

C Instructions
      instruction_1
      instruction_2
      ...
```

```
      end
C234567
```

3.3 Liens utiles

Quelques liens utiles concernant le Fortran :

- le *Professional Programmer's Guide to Fortran77* : http://www.star.le.ac.uk/~cgp/prof77.html
- les *FORTRAN Programming Language Resources around the World* : http://www.eeng.brad.ac.uk/help/.packlangtool/.langs/.fort/.resource.html
- le *Fortran Tutorial* de http://gershwin.ens.fr/vdaniel/Doc-Locale/Langages-Program-Scientific/Fortran/Tutorial/
- des supports de cours : http://dept-info.labri.u-bordeaux.fr/~billaud/travaux/IntroF77/intro-f77.html.

4 Premiers programmes

Exercice : Écrire, compiler et exécuter les programmes suivants. Les assaisonner à votre goût.

Entrée/Sortie. L'entrée et la sortie standards correspondent au clavier et à l'écran respectivement.

```
      PROGRAM pythagore
C Affiche z = sqrt(x**2+y**2), x et y entres au clavier
      IMPLICIT NONE
      REAL x,y,z

      PRINT*,"Entrer 2 nombres: "
      READ(*,*) x,y
      z = sqrt(x**2+y**2)
      WRITE(*,*) "hypothenuse=",z

      END
```

Fichiers. Pour écrire ou lire dans un fichier, il faut l'avoir ouvert au préalable, puis le fermer.

```
      PROGRAM sinus
C stocke une periode de sinusoide dans le fichier
C sinus.dat, la periode et le nb de points etant
```

```
C donnee par le fichier input.dat
      IMPLICIT NONE
      INTEGER in,out
      INTEGER n,i
      REAL periode,x,y
      DOUBLE PRECISION pi

C Ouverture d'un fichier en lecture
      PRINT*,"Lecture du fichier input.dat..."
      in = 1
      OPEN(in,file='input.dat',status='old')
      READ(in,*) periode
      READ(in,*) n
C Ouverture d'un fichier en ecriture
      PRINT*,"Creation du fichier sinus.dat..."
      out = 2
      OPEN(out,file='sinus.dat',status='unknown')

C Boucle sur une periode
      pi = 4.*atan(1.)
      PRINT*,"Pi =",pi
      DO i = 0,n
         x = i*periode/n
         y = sin(x/periode*2.*pi)
         WRITE(out,*) x,y
      END DO

C Fermer les fichiers
      PRINT*,"Fermeture des fichiers"
      CLOSE(in)
      CLOSE(out)

      END
```